**Міністерство освіти і науки України**

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра обчислювальної техніки**

**Лабораторна робота №6**

з дисципліни

«ООП»

на тему «Побудування програмної системи з множини об’єктів, керованих повідомленнями»

Виконав: Перевірив:

студент групи ІП-93 Порєв Віктор Миколайович

Завальнюк Максим Євгенович

номер залікової книжки: 9312

Київ 2020

Мета**:** отримати вміння та навички використовувати засоби обміну інформацією

та запрограмувати взаємодію незалежно працюючих програмних компонентів.

Зміст

[Мета 2](#_Toc58788627)

[Завдання 3](#_Toc58788628)

[1. Створити у середовищі MS Visual Studio C++ проект Win32 з ім’ям Lab6. 3](#_Toc58788629)

[2. Написати вихідний текст усіх програм-компонентів згідно варіанту завдання. 3](#_Toc58788630)

[3. Скомпілювати вихідні тексти і отримати виконуваний файли програм. 3](#_Toc58788631)

[4. Перевірити роботу програм. Налагодити взаємодію програм. 3](#_Toc58788632)

[5. Проаналізувати та прокоментувати результати та вихідні тексти програм. 3](#_Toc58788633)

[6. Оформити звіт. 3](#_Toc58788634)

[Варіанти завдань 3](#_Toc58788635)

[Вихідні тексти файлів 4](#_Toc58788636)

[Lab6.cpp 4](#_Toc58788637)

[Object1.cpp 8](#_Toc58788638)

[Object2.cpp 12](#_Toc58788639)

[Скріншоти програми 17](#_Toc58788640)

[Відображення головної програми 17](#_Toc58788641)

[Відображення програми Object1 та Object2 17](#_Toc58788642)

[Контрольні запитання 18](#_Toc58788643)

[Висновок 18](#_Toc58788644)

Завдання**:**

# Створити у середовищі MS Visual Studio C++ проект Win32 з ім’ям Lab6.

# Написати вихідний текст усіх програм-компонентів згідно варіанту завдання.

# Скомпілювати вихідні тексти і отримати виконуваний файли програм.

# Перевірити роботу програм. Налагодити взаємодію програм.

# Проаналізувати та прокоментувати результати та вихідні тексти програм.

# Оформити звіт.

# Варіанти завдань

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер варіанту** | **Програма Lab6** | **Програма Object2** | **Програма Object3** |
| 2 | 1. Користувач вводить значення **n**, **Min**, **Max** у діалоговому вікні. 2. Програма викликає програми Object2, 3 і виконує обмін повідомленнями з ними для передавання, отримання інформації. | 1. Створює вектор n Дробових (**double**) чисел у діапазоні **Min** – **Max** 2. Показує числові значення у декількох стовпчиках та рядках у власному головному вікні 3. Записує дані в Clipboard Windows у текстовому форматі | 1. Зчитує дані з Clipboard Windows 2. Виконує сортування масиву чисел і відображає його у декількох стовпчиках та рядках у власному головному вікні |

# Вихідні тексти файлів

## Lab6.cpp

#include "stdafx.h"

#include "Lab6.h"

#include "dialog.h"

#define MAX\_LOADSTRING 110

// Глобальные переменные:

HINSTANCE hInst; // текущий экземпляр

WCHAR szTitle[MAX\_LOADSTRING]; // Текст строки заголовка

WCHAR szWindowClass[MAX\_LOADSTRING]; // имя класса главного окна

// Отправить объявления функций, включенных в этот модуль кода:

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

extern int Dialog(HINSTANCE hInst, HWND hWnd);

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: Разместите код здесь.

// Инициализация глобальных строк

LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadStringW(hInstance, IDC\_LAB6, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_LAB6));

MSG msg;

// Цикл основного сообщения:

while (GetMessage(&msg, nullptr, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int) msg.wParam;

}

//

// ФУНКЦИЯ: MyRegisterClass()

//

// ЦЕЛЬ: Регистрирует класс окна.

//

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_LAB6));

wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_LAB6);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassExW(&wcex);

}

//

// ФУНКЦИЯ: InitInstance(HINSTANCE, int)

//

// ЦЕЛЬ: Сохраняет маркер экземпляра и создает главное окно

//

// КОММЕНТАРИИ:

//

// В этой функции маркер экземпляра сохраняется в глобальной переменной, а также

// создается и выводится главное окно программы.

//

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance; // Сохранить маркер экземпляра в глобальной переменной

HWND hWnd = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

//

// ФУНКЦИЯ: WndProc(HWND, UINT, WPARAM, LPARAM)

//

// ЦЕЛЬ: Обрабатывает сообщения в главном окне.

//

// WM\_COMMAND - обработать меню приложения

// WM\_PAINT - Отрисовка главного окна

// WM\_DESTROY - отправить сообщение о выходе и вернуться

//

//

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

int result;

switch (message)

{

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

switch (wmId)

{

case IDM\_LABBUTTON:

result = Dialog(hInst, hWnd);

break;

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

return DefWindowProcW(hWnd, message, wParam, lParam);

}

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

EndPaint(hWnd, &ps);

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProcW(hWnd, message, wParam, lParam);

}

return 0;

}

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}

static INT\_PTR CALLBACK Dialog(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

char numbers[10];

char minimum[10];

char maximum[10];

HWND hWndDataCreator;

COPYDATASTRUCT cds;

HWND hWndL1;

int limit = 200;

int limitNumbers = 64;

int wmId = LOWORD(wParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_DESTROY: {

hWndL1 = FindWindow("Object1", NULL);

if (hWndL1) PostMessage(hWndL1, WM\_DESTROY, (WPARAM)wParam, 0);

break;

}

case WM\_COMMAND:

switch (wmId) {

case IDCANCEL:

EndDialog(hDlg, 0);

return (INT\_PTR)TRUE;

break;

case IDB\_CONTINUE:

GetWindowText(GetDlgItem(hDlg, IDC\_N), numbers, sizeof(numbers));

GetWindowText(GetDlgItem(hDlg, IDC\_MIN), minimum, sizeof(minimum));

GetWindowText(GetDlgItem(hDlg, IDC\_MAX), maximum, sizeof(maximum));

if (numbers[0] == '\0' || minimum[0] == '\0' || maximum[0] == '\0') {

MessageBox(GetParent(hDlg), (LPCSTR)"Введіть непусті дані", (LPCSTR)"Помилка", MB\_OK);

}

else if (atoi(minimum) > limit || atoi(maximum) > limit || atoi(numbers) > limitNumbers || atoi(minimum) >= atoi(maximum)) {

MessageBox(GetParent(hDlg), (LPCSTR)"Введіть коректні дані. Перевищений ліміт одного із параметрів", (LPCSTR)"Помилка", MB\_OK);

}

else {

long parametres[3] = {atoi(numbers), atoi(minimum), atoi(maximum)};

cds.dwData = 1;

cds.cbData = 9;

cds.lpData = parametres;

hWndDataCreator = FindWindow("Object1", NULL);

if (hWndDataCreator == NULL) {

WinExec("Object1.exe", SW\_SHOW);

hWndDataCreator = FindWindow("Object1", NULL);

}

SendMessage(hWndDataCreator, WM\_COPYDATA, (WPARAM)GetParent(hDlg), (LPARAM)&cds);

}

break;

}

}

return (INT\_PTR)FALSE;

}

int Dialog(HINSTANCE hInst, HWND hWnd) {

return DialogBox(hInst, MAKEINTRESOURCE(IDD\_MODULE), hWnd, Dialog);

}

## Object1.cpp

#include "stdafx.h"

#include "Object1.h"

#include <vector>

#include <time.h>

using namespace std;

#define MAX\_LOADSTRING 110

// Глобальные переменные:

HINSTANCE hInst; // текущий экземпляр

WCHAR szTitle[MAX\_LOADSTRING]; // Текст строки заголовка

WCHAR szWindowClass[MAX\_LOADSTRING]; // имя класса главного окна

long numbers;

vector<float> numbersVector;

// Отправить объявления функций, включенных в этот модуль кода:

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

void OnCopyData(HWND, WPARAM, LPARAM);

void launchObjectThree(HWND);

int Clipboard(HWND, const char \*);

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: Разместите код здесь.

// Инициализация глобальных строк

LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadStringW(hInstance, IDC\_OBJECT1, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_OBJECT1));

MSG msg;

// Цикл основного сообщения:

while (GetMessage(&msg, nullptr, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int) msg.wParam;

}

//

// ФУНКЦИЯ: MyRegisterClass()

//

// ЦЕЛЬ: Регистрирует класс окна.

//

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_OBJECT1));

wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_OBJECT1);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassExW(&wcex);

}

//

// ФУНКЦИЯ: InitInstance(HINSTANCE, int)

//

// ЦЕЛЬ: Сохраняет маркер экземпляра и создает главное окно

//

// КОММЕНТАРИИ:

//

// В этой функции маркер экземпляра сохраняется в глобальной переменной, а также

// создается и выводится главное окно программы.

//

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance; // Сохранить маркер экземпляра в глобальной переменной

HWND hWnd = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

//

// ФУНКЦИЯ: WndProc(HWND, UINT, WPARAM, LPARAM)

//

// ЦЕЛЬ: Обрабатывает сообщения в главном окне.

//

// WM\_COMMAND - обработать меню приложения

// WM\_PAINT - Отрисовка главного окна

// WM\_DESTROY - отправить сообщение о выходе и вернуться

//

//

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

srand(time(NULL));

HWND hWndL2;

int width = 200;

int height = 700;

int startX = 141;

int startY = 40;

switch (message)

{

case WM\_CREATE:

{

SetWindowPos(hWnd, HWND\_BOTTOM, startX, startY, width, height, SWP\_DEFERERASE);

break;

}

case WM\_COPYDATA:

{

numbersVector.clear();

OnCopyData(hWnd, wParam, lParam);

launchObjectThree(hWnd);

break;

}

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

switch (wmId)

{

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

return DefWindowProcW(hWnd, message, wParam, lParam);

}

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

for (int i = 0; i < numbersVector.size(); i++) {

int number = numbersVector[i];

int count = 0;

while (number) {

number /= 10;

count++;

}

char buffer[100] = {};

sprintf\_s(buffer, "%f", numbersVector[i]);

TextOut(hdc, 10 + int(i / 32) \* 100, (i % 32) \* 20, buffer, count + 7);

}

EndPaint(hWnd, &ps);

}

break;

case WM\_DESTROY:

hWndL2 = FindWindow("Object2", NULL);

if (hWndL2) PostMessage(hWndL2, WM\_DESTROY, (WPARAM)wParam, 0);

PostQuitMessage(0);

break;

default:

return DefWindowProcW(hWnd, message, wParam, lParam);

}

return 0;

}

// Обработчик сообщений для окна "О программе".

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}

void OnCopyData(HWND hWnd, WPARAM wParam, LPARAM lParam) {

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

COPYDATASTRUCT \*cds;

cds = (COPYDATASTRUCT \*)lParam;

long \*p = (long \*)cds->lpData;

numbers = p[0];

int minimum = p[1];

int maximum = p[2];

for (int i = 0; i < numbers; i++) {

numbersVector.push\_back(((maximum - minimum) \* ((float)rand() / RAND\_MAX)) + minimum);

}

InvalidateRect(hWnd, NULL, TRUE);

}

void launchObjectThree(HWND hWnd) {

HWND hWndDataCreator = FindWindow("Object2", NULL);

string str = "";

for (int i = 0; i < numbersVector.size(); i++) {

str += (to\_string(numbersVector[i]) + "\n");

}

char \*a = new char[str.size() + 1];

a[str.size()] = 0;

memcpy(a, str.c\_str(), str.size());

Clipboard(hWnd, a);

if (hWndDataCreator == NULL) {

WinExec("Object2.exe", SW\_SHOW);

}

else {

PostMessage(hWndDataCreator, IDB\_REBOOT, 0, 0);

}

}

int Clipboard(HWND hWnd, const char \*src) {

HGLOBAL hglbCopy;

BYTE \*pTmp;

long length;

if (src == NULL || src[0] == 0) return 0;

length = strlen(src);

hglbCopy = GlobalAlloc(GHND, length + 1);

if (hglbCopy == NULL) return 0;

pTmp = (BYTE \*)GlobalLock(hglbCopy);

memcpy(pTmp, src, length + 1);

GlobalUnlock(hglbCopy);

if (!OpenClipboard(hWnd)) {

GlobalFree(hglbCopy);

return 0;

}

EmptyClipboard();

SetClipboardData(CF\_TEXT, hglbCopy);

CloseClipboard();

return 1;

}

## Object2.cpp

#include "stdafx.h"

#include "Object2.h"

#include <vector>

#include <algorithm>

using namespace std;

#define MAX\_LOADSTRING 110

// Глобальные переменные:

HINSTANCE hInst; // текущий экземпляр

WCHAR szTitle[MAX\_LOADSTRING]; // Текст строки заголовка

WCHAR szWindowClass[MAX\_LOADSTRING]; // имя класса главного окна

vector<float> numbersVector;

// Отправить объявления функций, включенных в этот модуль кода:

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

long Clipboard(HWND, char \*, long);

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: Разместите код здесь.

// Инициализация глобальных строк

LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadStringW(hInstance, IDC\_OBJECT2, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_OBJECT2));

MSG msg;

// Цикл основного сообщения:

while (GetMessage(&msg, nullptr, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int) msg.wParam;

}

//

// ФУНКЦИЯ: MyRegisterClass()

//

// ЦЕЛЬ: Регистрирует класс окна.

//

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_OBJECT2));

wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_OBJECT2);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassExW(&wcex);

}

//

// ФУНКЦИЯ: InitInstance(HINSTANCE, int)

//

// ЦЕЛЬ: Сохраняет маркер экземпляра и создает главное окно

//

// КОММЕНТАРИИ:

//

// В этой функции маркер экземпляра сохраняется в глобальной переменной, а также

// создается и выводится главное окно программы.

//

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance; // Сохранить маркер экземпляра в глобальной переменной

HWND hWnd = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

//

// ФУНКЦИЯ: WndProc(HWND, UINT, WPARAM, LPARAM)

//

// ЦЕЛЬ: Обрабатывает сообщения в главном окне.

//

// WM\_COMMAND - обработать меню приложения

// WM\_PAINT - Отрисовка главного окна

// WM\_DESTROY - отправить сообщение о выходе и вернуться

//

//

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

char text[700] = {};

string str;

int width = 200;

int height = 700;

int startX = 326;

int startY = 40;

switch (message)

{

case WM\_CREATE:

{

SetWindowPos(hWnd, HWND\_NOTOPMOST, startX, startY, width, height, SWP\_DEFERERASE);

}

case IDR\_REBOOT:

{

numbersVector.clear();

Clipboard(hWnd, text, 700);

str = text;

string number;

while (str != "") {

number = str.substr(0, str.find\_first\_of("\n"));

char buffer[100] = {};

numbersVector.push\_back(stod(number));

str = str.substr(str.find\_first\_of("\n") + 1);

}

sort(numbersVector.begin(), numbersVector.end());

InvalidateRect(hWnd, NULL, TRUE);

}

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

switch (wmId)

{

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

return DefWindowProcW(hWnd, message, wParam, lParam);

}

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

for (int i = 0; i < numbersVector.size(); i++) {

int num = numbersVector[i];

int count = 0;

while (num) {

num /= 10;

count++;

}

char buffer[100] = {};

sprintf\_s(buffer, "%f", numbersVector[i]);

TextOut(hdc, 10 + int(i/32) \* 100, (i % 32) \* 20, buffer, count + 7);

}

EndPaint(hWnd, &ps);

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProcW(hWnd, message, wParam, lParam);

}

return 0;

}

// Обработчик сообщений для окна "О программе".

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}

long Clipboard(HWND hWnd, char \*dest, long maxsize) {

HGLOBAL hglb;

LPTSTR lptstr;

long size, result;

result = 0;

if (!IsClipboardFormatAvailable(CF\_TEXT) || !OpenClipboard(hWnd)) return 0;

hglb = GetClipboardData(CF\_TEXT);

if (hglb != NULL) {

lptstr = (char \*)GlobalLock(hglb);

if (lptstr != NULL) {

size = strlen(lptstr);

strcpy\_s(dest, maxsize, lptstr);

result = size;

GlobalUnlock(hglb);

}

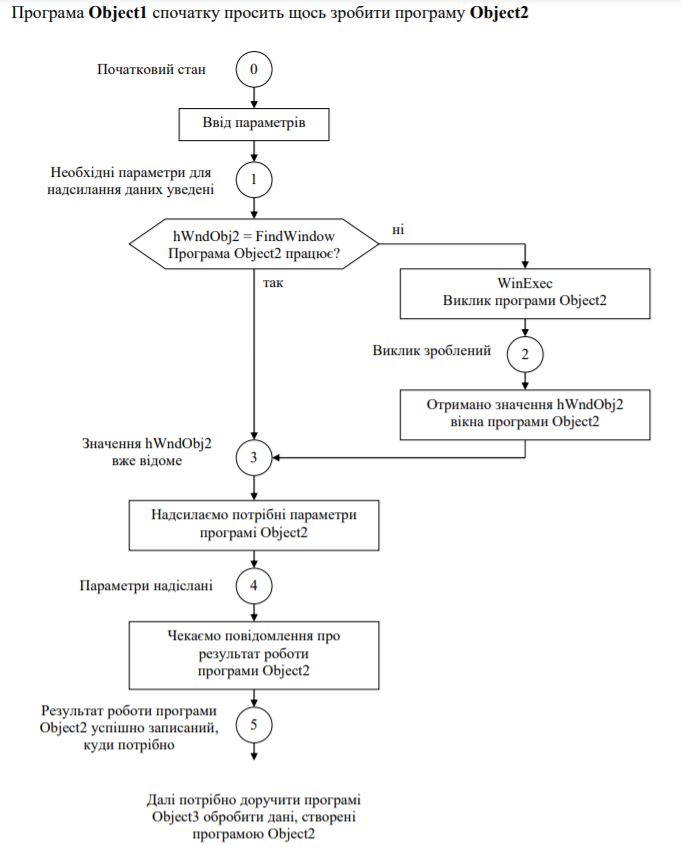
}

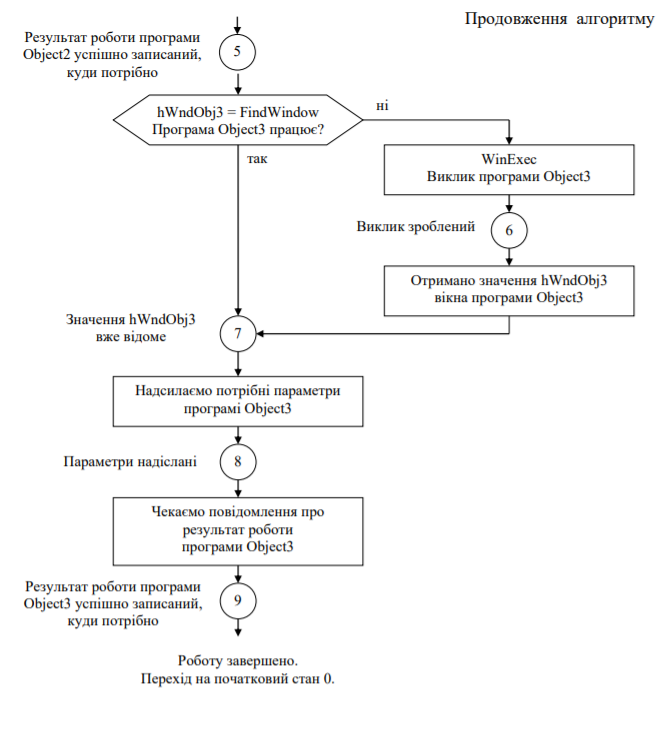
CloseClipboard();

return result;

}

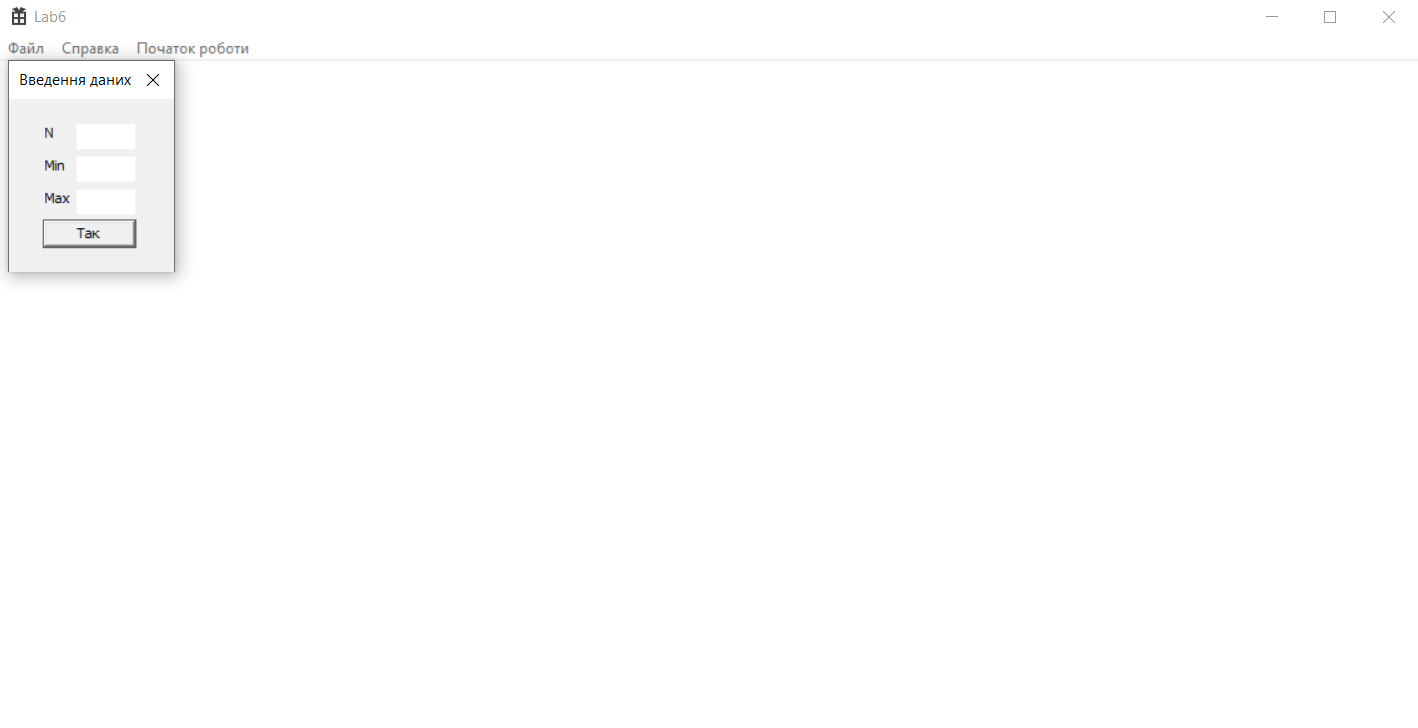
Схема послідовності надсилання-обробки повідомлень



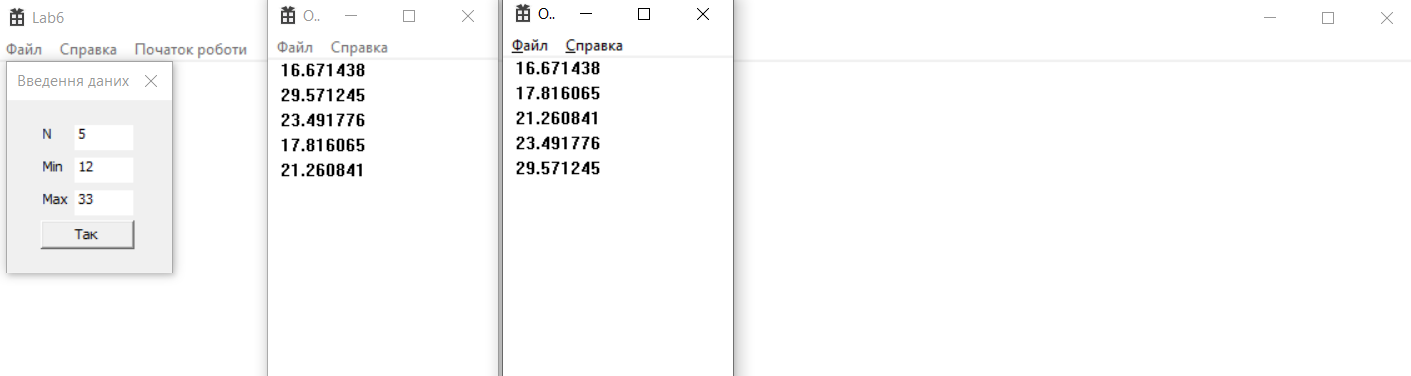


# Скріншоти програми

## Відображення головної програми



## Відображення програми Object1 та Object2



# Контрольні запитання

1. Як запрограмувати виклик у програмі іншої програми?

Для того, щоб запрограмувати виклик іншої програми, потрібно використати вбудовану функцію *WinExec*. У неї такий синтаксис: WinExec("otherprog.exe", SW\_SHOW), де "otherprog.exe" - це назва програми, яку необхідно викликати.

1. Як отримати hWnd іншої програми?

Достатньо застосувати ту ж саму вбудовану функцію *WinExec*. Тільки на цей раз разом з іменем файлу через пробіл надіслати **hWnd** відповідної програми.

1. Як надіслати іншій програмі повідомлення?

За допомогою функцій *SendMessage* або *PostMessage*. У наступних питання вони обидві будуть розглянуті.

1. Як надіслати іншій програмі одне число?

Наприклад, за допомогою функції *PostMessage*. Ця функція, на відміну від *SendMessage*, яка надсилає повідомлення **позачергово**, записує повідомлення у **чергу**.

1. Як надіслати іншій програмі масив числових значень?

Наприклад, за допомогою функції *SendMessage*. Також нам знадобиться Windows-повідомлення **WM\_COPYDATA**, яке зможе обробити надіслане повідомлення.

1. Як запрограмувати обмін інформацією з Clipboard Windows?

Для реалізації **Clipboard Windows** використовується *глобальна динамічна віртуальна пам’ять*. Рекомендується використовувати Clipboard для обміну даними обсягом не більше декількох десятків мегабайтів. Для того, щоб програма записала дані у **Clipboard**, потрібно викликати функцію API Windows *SetClipboardData*. Ця функція має у якості параметрів ідентифікатор формату та handle блоку пам’яті, відкритого за допомогою функцій GlobalAlloc.

А щоб приймати дані через повідомлення WM\_COPYDATA, можна використати поле cds->dwData для ідентифікації кількості та типу даних, що надсилаються-отримуються.

# Висновок

У ході виконання лабораторної роботи я отримав вміння та навички у використанні засобів обміну інформації, навчився програмувати незалежно працюючі програмні компоненти та детально ознайомився з ними на практиці.